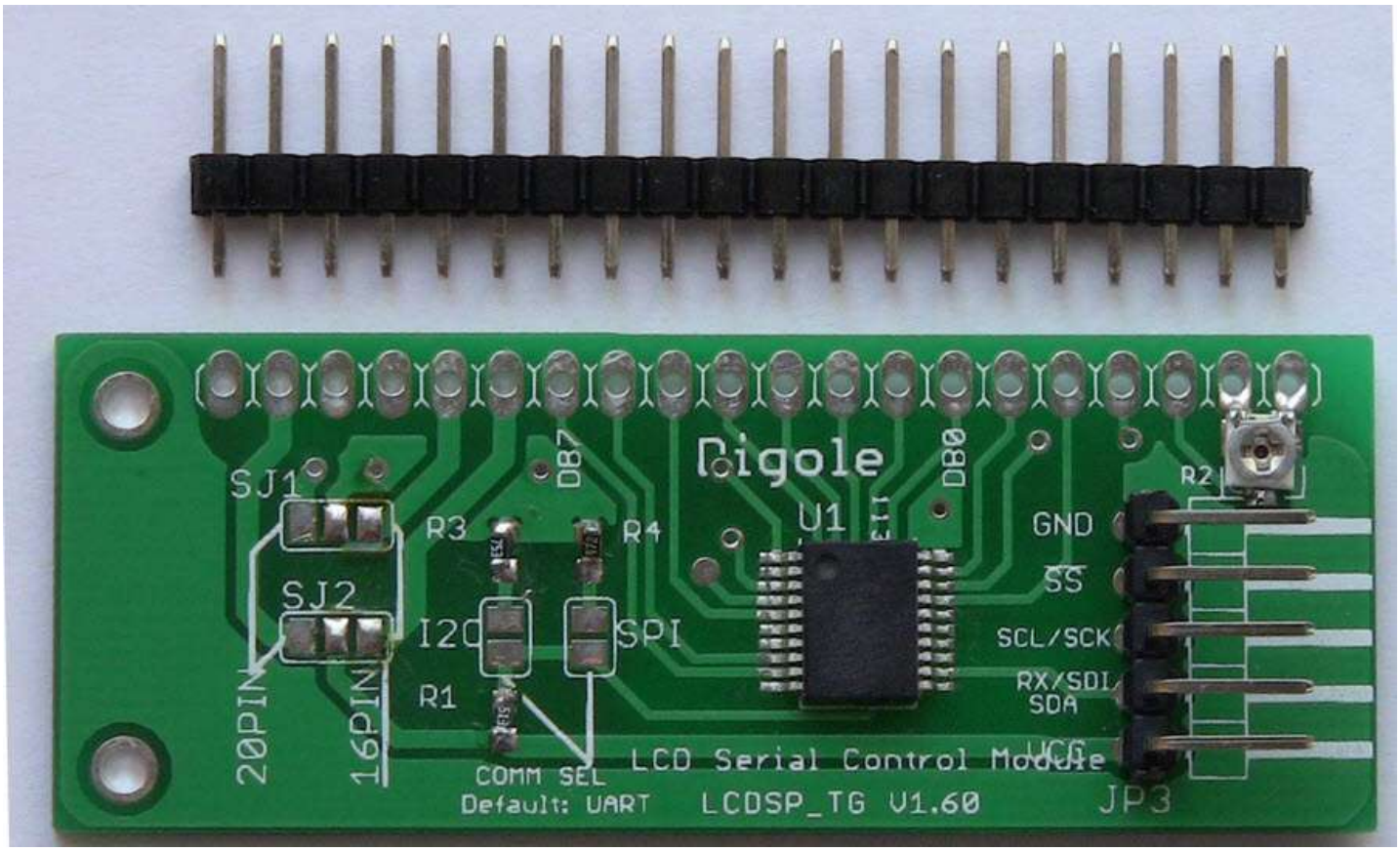
Serial:UART/I2C/SPI Text LCD Display Control Module

LCDSP_TG V1.60 user manual

Product picture:



In order to use a Text LCD display, you need connect at least 4 data pins and 2 control pins from your circuit to display, and you also need to spend some time to write code to drive it, in some case, you may need couple of days to deal with it, and will give you a nightmale when changing to other display.

This control module (adapter) will help you to solve this issue, it can work with most 1602/1604/2002/2004 LCD that config with KS0066U/F / HD44780 and other compatible chips seemless, so you can use your dispaly with this module in couple of minutes!

Tthe module can be set up as UART or I2C or SPI mode that communicate with your circuit by short the jumper on board, and the UART BAUD and I2C address can be set up by yourself also (the module don't store the BAUD, so it alway be 9600 after reset, but I2C address will be stored in module).

This module can connect with16pins and 20pins LCD interface, by set the jumper of SJ1 and SJ2 on board.

FEATURES:

- Supply voltage: 1.8V to 5.5V
- Communication mode: UART/I2C/SPI, detect your setting automatically
- Receiving buffer: 64 bytes
- Work with KS0066U/F / HD44780 and other compatible chips
- Work with most 1602/1604/2002/2004 LCD module
- Low power consumption: 4mA maximum
- Simple command sets, easy to remember
- Default setting: UART baud 9600bps, I2C 0x27 address
- UART baud (bps): 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 115200
- Manual, Arduino lib, sample code download here

When you need display text on LCD by using the module, just send the following command to module through UART, I2C or SPI ("x" means a byte, "..." means varible bytes):

IMPORTANT: due to 0 is the end char of a string, and the display position start at x=0, y=0, so, if you need set a postion of 0, you can't put the 0 in a string, you need send this byte individually.

Character Display Command: (B-one byte, B…-Bytes)

Command	Description	Arduino lib function	note
CL	CLear screen and set the display position to first Column and first Row (x=0.y=0)	clearScreen();	
CSB	set CurSor on/off	enableCursor(); disableCursor();	
DCB	Display Config on/off, the factory default set is on, so, when the module is powered up, it will display current communication mode on LCD, after you design finished, you can turn it off	displayConfig(0); displayConfig(1);	
SBB…	Set UART Baud, B are ASCII characters, the available values are: “300”, “1200”, “2400”,”4800”, “9600”, “14400”, “19200”, “28800”, “38400”, “57600”,”115200”	Set BAUT when initial the class	When adapter power up or reset, always start with 9600 Baud
SI2CAB	Set I2C Address,the default address is 0x27, the adapter will store the new address in memory	setI2CAddress(0x34);	Change address to 0x34
STCRBB BBBB	Set Text Columns and Rows, this command will config your LCD if other than 1602 and the chip is other than KS0066U/F / HD44780	setLCDColRow(20,4);	The last 4 B should be“\x80\xC0\x94\xD4”
TPBB	set Text Position for following display	setPrintPos(6,1);	This will affect the following “TT” command

TTB…	display TexT string, the text will wraped in next row if the current row fulled, the Text Postion will be changed to the last char displyed, this command ended by 0x00 or 0x0D received	print(string);
		print(int);
		print(char);
		print(float);
		print(double);
		drawStr(x,y,string);
			The print function in Arduino, also can print other data and format the out put, please refer to Serial.print.

Pinout of module connect to LCD display:

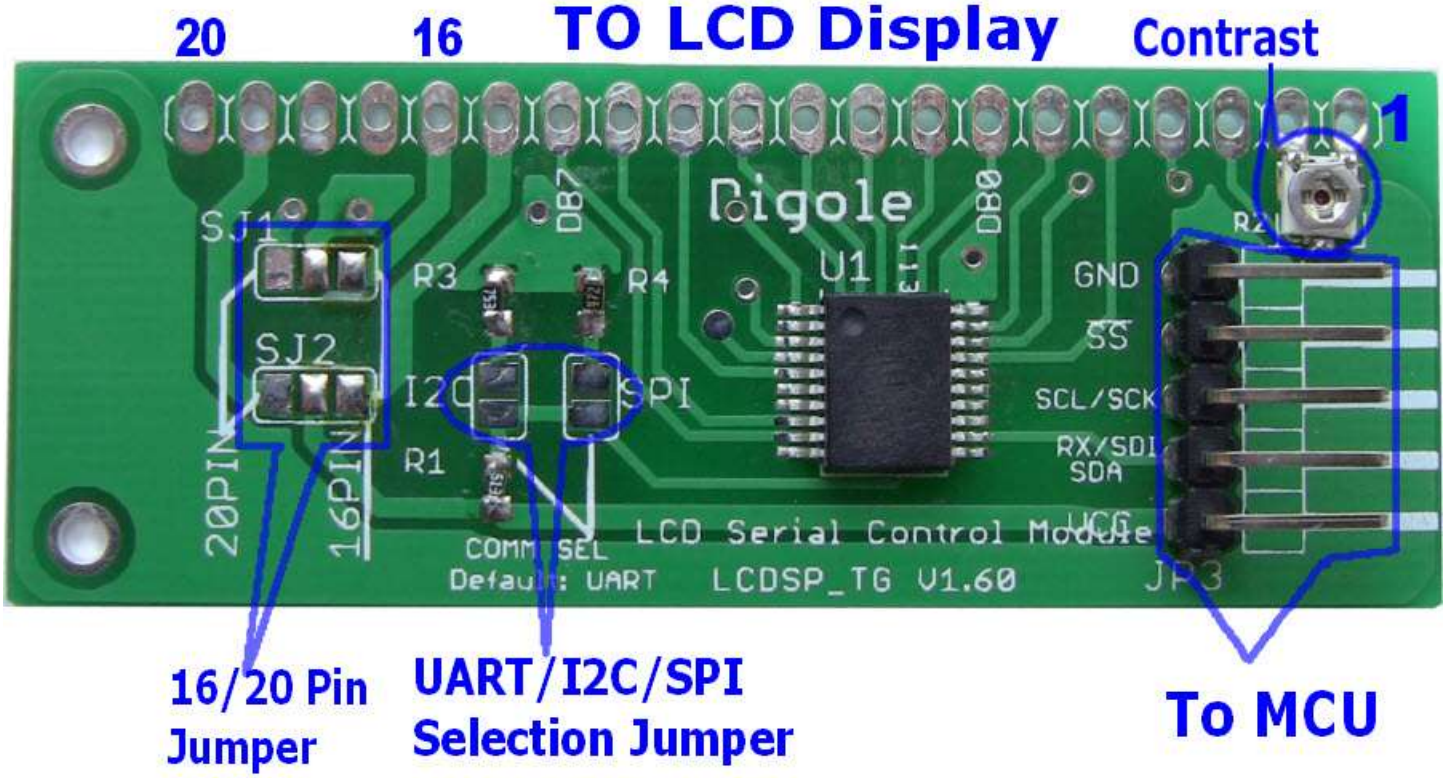
PIN	Description	PIN	Description
1	VSS Ground of circuit	2	VDD Power Supply
3	V0	4	RS
5	R/W	6	E
7	DB0 or D0	8	DB1 or D1
9	DB2 or D2	10	DB3 or D3
11	DB4 or D4	12	DB5 or D5
13	DB6 or D6	14	DB7 or D7
15	**16pin mode:** BLA or A, + power supply for backlit	16	**16pin mode:** BLK or K, - power supply for backlit

PIN	Description	PIN	Description
	20pin mode: N/A		20pin mode: N/A
17	**20pin mode only:** RST:System reset input (low active).	18	**20pin mode only:** VOUT
19	**20pin mode only:** BLA or A, + power supply for backlit	20	**20pin mode only:** BLK or K, - power supply for backlit

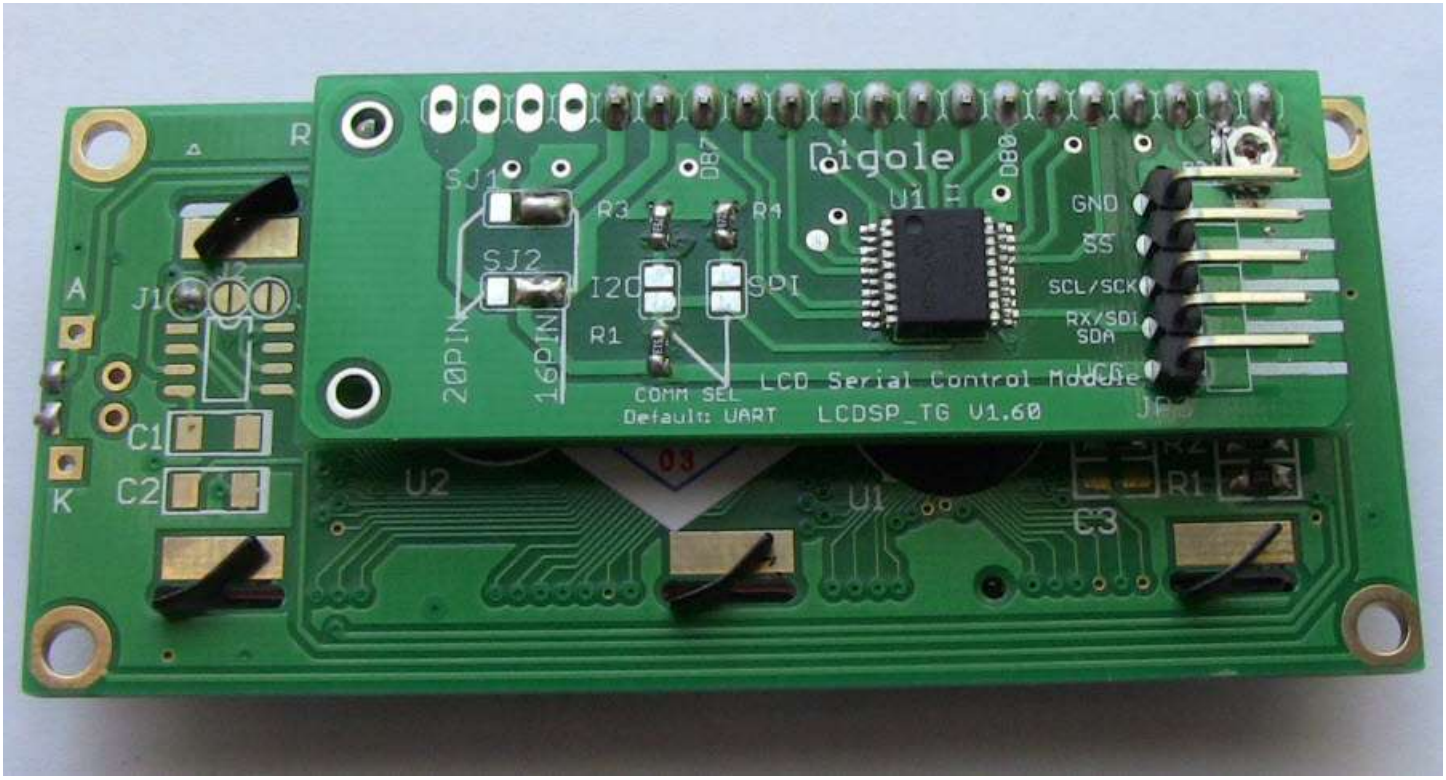
Pinout of this module connect to MCU:

PIN	Description	PIN	Description
1	GND (0V)	2	SS: **SPI mode only** chip select control in, low active
3	**I2C and SPI mode:** SCK/SCL: Clock in	4	**UART mode:** RX **I2C mode:** SDA **SPI mode:** SDI
5	VCC: power supply 1.8V to 5.5V depends on you LCD		

Config your module:



Work with a 1602:



Arduino Sample Code:

```
/*----------NOTE----------
new version of lib will save you compiled code size for the sample
The size for 2 version:
    UART  I2C    SPI
OLD   8998  8988   9132
NEW   6966  7566   6354
------------------------*/
#define _Digole_Serial_UART_  //To tell compiler compile the special communication only,
//other available is: _Digole_Serial_I2C_ and _Digole_Serial_SPI_
#include <DigoleSerial.h>
//--------UART setup, if you don't use UART, use // to comment following line
DigoleSerialDisp mydisp(&Serial, 9600); //UART:Pin 1(TX)on arduino to RX on module
//--------I2C setup, if you don't use I2C, use // to comment following 2 lines
//#include <Wire.h>
//DigoleSerialDisp mydisp(&Wire,'\x27');  //I2C:SDA (data line) is on analog input pin 4, and SCL (clock line) is on analog input pin 5
//--------SPI setup, if you don't use SPI, use // to comment following line
//DigoleSerialDisp mydisp(8,9,10);  //SPI:Pin 8: data, 9:clock, 10: SS, you can assign 255 to SS, and hard ground SS pin on module
#define LCDCol 16
#define LCDRow 2

int ptr;
const char a[] = "disp char array";
const char b = 'Q';
int c = 3456;
String d = "I'm a string";
float pi = 3.1415926535;
double lg10;

void setup() {
  mydisp.begin();
  /*----------for text LCD adapter and graphic LCD adapter ----------*/
  mydisp.clearScreen(); //CLear screen
  //mydisp.displayConfig(1);    //set config display ON, 0=off
  //mydisp.setI2CAddress(0x29);  //this function only working when you connect using I2C, from 1 to 127
  //delay(1000);
  //mydisp.setLCDColRow(16,2);  //set LCD Col and Row, only time set up is OK
  mydisp.disableCursor(); //disable cursor, enable cursore use: enableCursor();
  mydisp.drawStr(4, 0, "Demo now"); //display string at: x=4, y=0

  //Test print function
  mydisp.setPrintPos(0, 1);
  mydisp.print(a); // display a char array
```

```
resetpos();
mydisp.print("display a char:");
mydisp.print(b); //display a char
resetpos();
mydisp.print("int as DEC:");
mydisp.print(c); //display 3456 in Dec
resetpos();
mydisp.print("as HEX:");
mydisp.print(c, HEX); //display 3456 in Hex
resetpos();
mydisp.print("as OCT:");
mydisp.print(c, OCT); //display 3456 in Oct
resetpos();
mydisp.print("BIN:");
mydisp.print(c, BIN); //display 3456 in Bin
resetpos();
mydisp.print(d); //display String object
resetpos();
mydisp.print("float pi=");
mydisp.print(pi); //display float of PI
resetpos();
mydisp.print("Pi6=");
mydisp.print(pi, 6); //display PI in 6 Accuracy
resetpos();
mydisp.print("Pi*3=");
mydisp.print(pi * 3, 6); //display PI time 3 in 6 Accuracy
resetpos();
mydisp.print("lg5=");
mydisp.print(log(5), 8); //display log(5) in 8 Accuracy
resetpos();
for (uint8_t j = 0; j < 4; j++) //making "Hello" string moving
{
  for (uint8_t i = 0; i < 10; i++) //move string to right
  {
    mydisp.setPrintPos(i, 1);
    mydisp.print(" Hello ");
    delay(100); //delay 100ms
  }
  for (uint8_t i = 0; i < 10; i++) //move string to left
  {
    mydisp.setPrintPos(9 - i, 1);
    mydisp.print(" Hello ");
    delay(100);
  }
}
mydisp.print("Enjoy it!");
```

```
   mydisp.enableCursor(); //enable cursor
}

void resetpos(void) //for demo use, reset display position and clean the demo line
{
   mydisp.setPrintPos(0, 1);
   delay(2000); //delay 2 seconds
   mydisp.println("              "); //display space, use to clear the demo line
   mydisp.setPrintPos(0, 1);
}
void loop() {
}
```